https://doi.org/10.2352/ISSN.2470-1173.2019.11.IPAS-279

Modeling lens optics and rendering virtual views from fisheye imagery

Filipe Gama, Mihail Georgiev, Atanas Gotchev
Tampere University, Tampere, Finland

Abstract

Production of high-quality virtual reality content from real sensed data is a challenging task due to several factors such as calibration of multiple cameras and rendering of virtual views. In this paper, we present a pipeline that maximizes the performance of virtual view rendering from an imagery captured by a camera equipped with fisheye lens optics. While such optics offer a wide field-of-view, it also introduces specific distortions. These have to be taken into account while rendering virtual views for a target application (e.g., head-mounted displays). We integrate a generic camera model into a fast rendering pipeline where we can tune intrinsic and extrinsic camera parameters along with resolution to meet the device or user requirements. We specifically target CPU-based implementation and quality in par with GPU-based rendering approaches. Using the adopted generic camera model, we numerically tabulate the required backward projection mapping and store it in a look-up table. This approach offers a trade-off between memory and computational complexity in terms of operations for calculating the mapping values. Finally, we complement our method with an interpolator, which handles occlusions efficiently. Experimental results demonstrate the viability, robustness and accuracy of the proposed pipeline.

Introduction

In recent years, Virtual Reality (VR) has attracted the attention of the general public and scientific community due to its ability to deliver immersive and interactive experiences like never before. Nowadays, VR content may be experienced in many areas, including entertainment, healthcare, engineering, education, military training, flight simulation and therapy. What makes VR unique is that the user has the feeling of "presence" inside the content. Thus, VR is often characterized based on the number of Degrees-of-Freedom (DoF) provided to the final user, which are typically 6-DoF or 3-DoF. In 6-DoF VR experience, the user has the possibility to move freely in a virtual space volume [1, 2, 3]. In this case, 3-DoF corresponds to the rotational motion of the user and the other 3-DoF to the translational motion. In similar fashion, 3-DoF VR expirience refers to rotational motion in which the user can only look around from the same viewpoint (a.k.a. omnidirectional VR system) [4]. On top of that, there are other factors that are equally important when it comes to deliver immersive experiences. One of them is deeply connected to human visual system [5]. In short, VR requires display systems (e.g., head-mounted display) that can provide realistic depth cues, spatial resolution, and field-of-view to meet human's eye sensitiveness and resolution power. Moreover, the overall display system should also be responsive to accommodate human's head or body motion. Another key factor in VR concerns the capturing setup or content production. VR content can either be fully computer-generated (explicit geometry), image-based (no geometry) or a mixture of these two (implicit geometry). In image-based capturing systems [6], the output VR content is often represented in the form of panoramas, concentric mosaics, or light fields and variants of these. This paper aims at studying conventional VR systems intended to capture real scenes through multiple cameras with wide-angle lenses (a.k.a. fisheye lenses) in order to serve any of these formats.

The conventional way of capturing a surrounding panorama for VR applications is either by rotating a camera around a single point [4, 6] or by using a 360-degree camera rig, like in [7]. To reduce the amount of cameras, capture non-static scenes, and in many cases achieve real-time performance, the scene can be captured through cameras with wide Field-of-View (FoV), i.e., cameras with wide-angle lenses. However, such solution has two main drawbacks. Firstly, wide-angle lenses impose significant optical aberrations. In this paper we consider only image distortions, which are taken into account digitally, after image acquisition. Secondly, cameras with wide-angle lenses impose a trade-off between angle-of-view and sensor resolution. The scene information gathered by a camera with wide-angle lens tends to be squeezed near the edges of the respective fisheye image. In other words, fisheye images are the result of a non-uniform sampling of the captured content so that it can fit in a reasonable camera sensor size.

In terms of distortions, typically, only two components are considered: radial and tangential or decentering distortions. However, since the predominant distortion component of fisheye lenses is radial distortion, and due to the manufacturing quality of today's cameras, tangential distortion is often omitted in the literature. Nevertheless, modulation and compensation for these distortions are taken into account in the so-called camera model. It is upon this integration that most methods of the literature diverge because the camera model can be interpreted from different perspectives. One option is to use the pinhole model as a base to project a 3D point onto the image plane, and then apply a non-linear distortion function in order to obtain the respective distorted point [8, 9, 10]. Another way of tackling the problem is to consider captured rays that impinge the lens. A relation is established between the incident rays direction defined from a 3D point to the optical center of the lens, and the distance between the projected point and the principal point [11]. Both cases stated above start by defining the forward projection function, i.e. how a 3D point is mapped onto the camera sensor. However, one can also define the camera model by starting from the camera sensor side. In this

case, the camera model starts by defining the backward projection function [12]. Pinhole-based approaches work up to a certain FoV, whereas in ray-based approaches this constraint may not exist. This particular observation is taken into account in this paper since we consider wide-angle lenses that can reach a FoV value greater than π rad.

In terms of visualization of captured VR content, many applications require (semi-)undistorted images. For instance, in Head-Mounted Displays (HMDs) or virtual reality headsets, the light passing through the device's lenses gets distorted and this has to be considered when displaying images [13]. In the conventional VR rendering pipeline from fisheye images, the images are first projected and stitched onto an unwrapped 3D surface (e.g., sphere, cylinder, cube). This unwarpped surface (or 2D plane), referred to as panorama, is then used as a texture map, a technique widely used in computer graphics to render images around a 3D object. Despite the functionality of this pipeline, there are few drawbacks. Firstly, this pipeline requires backward projection of the fisheye image data, which can be problematic when dealing with non-linear functions [12]. Secondly, this is an extensive pipeline that requires heavy computational resources. However, it cases which do not explicitly require a panorama, it can be simplified. Thus, another goal of this paper is to demonstrate how a generic camera model can be applied efficiently to render desired views in this context.

This paper is organized as follows: Section 2 introduces the chosen fisheye camera model and justifies the use of a ray-based model instead of a pinhole-based model. In Section 3 and 4, we describe the procedure to simulate fisheye cameras and render an arbitrary virtual view from it. Then, in Section 5, we present our experiments and results. Finally, conclusions are given in Section 6.

Fisheye camera model

Cameras equipped with fisheye lenses can cover a large FoV. In some cases, this type of lenses may achieve FoV values greater than π rad, introducing severe image distortions namely radial distortions. In the literature, most camera models are not ready to handle such wide angles. Thus, the aim of this section is to describe and stretch the importance of working with a generic ray-based model to model any lens type, from wide to ultra-wide angles.

The fisheye model considered in this paper is based on a radially symmetric ray-based model [11] with a small modification in the distortion coefficients value. The model takes into account the directionality of captured rays, expressed through a relation between the incident angle and the fisheye image radius. For the sake of simplicity, we illustrate the projection of a 3D world point **P** in camera coordinates onto a 2D image point **p** in Figure 1. For instance, in pinhole camera model, the distance r in pixels between an image point **p'** and the principal point O increases with the angle θ of the incoming ray:

$$r(\theta) = f\tan(\theta)\,, \tag{1}$$

where f is the focal length. This projection is valid for $\theta \in [0, \pi/2)$ rad. When θ approaches $\pi/2$ rad, a projected 3D point is infinitely far from the principal point ($r \to \infty$). Therefore, fisheye lenses require more sophisticated models that are either classified as pinhole-based models or ray-based models.

Pinhole-based models are widely used in calibration of fisheye

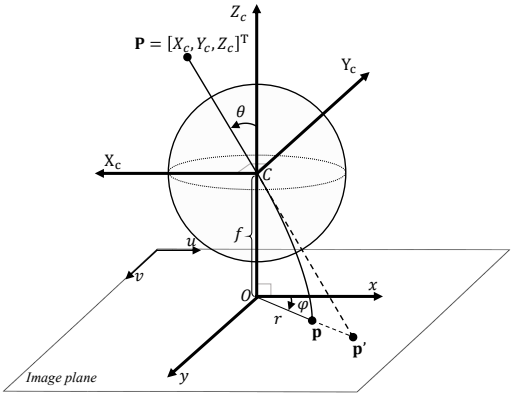


Figure 1. *Projection of a 3D point **P** onto a 2D image point **p** (polynomial model) and **p'** (pinhole model). The incident angle θ is defined between the optical axis and the incoming ray, and φ is given by the x-axis and the distance r. C is the camera optical centre, O represents the principal point, f is the focal length, and r is the distance between the principal point and a projected point.*

lenses with radial symmetry [9, 10]. In this group of models, a 3D world point is first projected onto the image plane using pinhole model. Then, the projected point is transformed according to some distortion function [4]. In the literature, one can find different functions to accomplish this task but essentially, their goal remains the same: minimize the amount of parameters required by the model (a.k.a. projection parameters, calibration parameters or distortion coefficients) and be effective against strong radial distortions. Since these models are built on top of pinhole model, they are limited in terms of projected rays with high incident angle (θ) values. On the other hand, ray-based models are more general and less limited. The idea is to cast the ray and establish a mapping function (or forward projection function) that relates θ with distance r. Like in the previous case, there are different models including equirectangular, equisolid angle, orthogonal, equidistance, and others [11, 12]. Equirectangular, equisolid angle, orthogonal and equidistance models use trigonometrical functions or simple functions with almost no parameters. Therefore, they are suitable for cameras with limited distortions and limited FoV. Alternatively, polynomial functions proposed in [11, 12] are able to address strong distortions and model any kind of lens. Polynomial functions used for radial distortions have been tested along with other methods mentioned above. Some authors have shown that this particular type of functions are more effective against strong distortions [14, 15]. In [11], the authors use a polynomial function to map the rays onto the camera sensor. The same concept is used in [12] but in inverse order i.e., mapping from camera sensor to ray.

In our paper, we consider a generic model, similar to [11]:

$$r(\theta) = f(m_0\theta + m_1\theta^3 + m_2\theta^5 + ... + m_n\theta^{2n+1})\,, \tag{2}$$

where $\{m_0, m_1, ..., m_n\}$ are the projection parameters or distortion coefficients and $n \in \mathbb{Z}_{\geq 0}$. In this polynomial function and unlike in [11], we assume $m_0 = 1$ to distinguish focal length from distortion coefficients, which is convenient for calibration purposes.

This assumption has a marginally positive impact in terms of accuracy compared to the original one as shown in Section 5. In general, the accuracy of this model increases with the number of parameters. The authors in [11] suggest that a ninth order polynomial provides sufficient DoF to approximate a variety of ray-based models. The mapping of the incoming ray to the Cartesian coordinates $(x,y)^{\mathrm{T}}$ is defined as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = r(\theta) \begin{bmatrix} \cos(\varphi) \\ \sin(\varphi) \end{bmatrix} , \tag{3}$$

where θ and φ are the angles related to the direction of the incoming ray (Figure 1). In terms of Cartesian coordinates, both angles can be expressed by:

$$\theta = \arccos\left(\frac{Z_c}{\sqrt{X_c^2 + Y_c^2 + Z_c^2}} \right) , \tag{4}$$

and

$$\varphi = \begin{cases} \arcsin\left(\frac{Y_c}{\sqrt{X_c^2 + Y_c^2}} \right) = \arccos\left(\frac{X_c}{\sqrt{X_c^2 + Y_c^2}} \right) \\ 0 \leftarrow \sqrt{X_c^2 + Y_c^2} = 0 \end{cases} . \tag{5}$$

In some works across the literature and available calibration software such as OpenCV, θ is expressed using tangent function. This may limit the model or add an extra degree of complexity because one must choose the correct quadrant of the Euclidean plane to obtain the correct θ value. Therefore, we ensure, by using Equation (4), that $\theta \in [0, \pi]$ rad. Finally, the pixel coordinates $(u,v)^{\mathrm{T}}$ are obtained as follows

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} O_x \\ O_y \end{bmatrix} , \tag{6}$$

with $(O_x, O_y)^{\mathrm{T}}$ being the coordinates of the principal point. This model assumes radial symmetry but in practice real lenses may deviate from it due to flaws in the optical elements. Therefore, Equation (3) can be complemented with asymmetric radial and tangential distortion components in a generic form as explained in [11] or by adapting other models like the classical Brown-Conrady model [16].

From the point of view of practical applications such as calibration, a camera model must describe both forward and backward projections. A high order polynomial model is effective against high radial distortions however, there is no analytic formula to compute its inverse. This is considered its main drawback compared to other projection functions mentioned above. Nevertheless, the backward projection may be computed through numerical methods [12], complex mapping functions [11], or Look-Up Table (LUT) for more accurate results. The third option is more convenient for the purposes of our fisheye lens simulator described in the next section. The role of LUT is to store angular information about the forward projection function, specifically $r(\theta)$, $\cos(\varphi)$ and $\sin(\varphi)$ information. In the same fashion, we could in principle store additional information regarding asymmetric distortions. Thanks to LUTs, one can easily convert projected points into ray information no matter the complexity of our forward projection function. This approach offers a trade-off between memory and computational complexity in terms of operations required to compute the mapped values.

Fisheye camera simulation

In the previous section, we discussed the advantages of utilizing a generic ray-based model over a pinhole-based model. The aim of this section is to describe a fisheye camera simulator because single dataset images are often not enough to validate or compare different camera models, or even evaluate our virtual view rendering approach described in the next section. In the proposed simulator pipeline, the main goal is to import any camera setup (e.g., 360-degree camera rig) and then, for each camera position, simulate the fisheye effect according to a certain camera model.

For the simulation of fisheye images, we make use of a third-party rendering software (e.g., Blender) to extract color and respective depth (Z-buffer) information. This information is sufficient to characterize any point in 3D space, its location and respective color. The full pipeline of the developed simulator is depicted in Figure 2. The first stage is dedicated to import information about the capturing system into the rendering software. This includes orientation and location of each individual camera in space. Then, for each camera, densely overlapped perspective views are rendered (e.g., 9 perspective images) by purely rotating the camera around its optical center. The result is shown in Figure 2-b). The goal of this stage is to gather information around the camera within a 360-degree FoV. In conventional rendering systems, each rendered pixel takes into account the same amount of rays. In other words, there is an uniform distribution of rays per pixel across the rendered image. In our pipeline this is not possible. Thus, we render a set of very high-resolution perspective images from the rendering software in order to ensure that in the worst case scenario a rendered pixel is generated from a significant set of rays to avoid resampling problems. These perspective views follow Equation (1) and therefore, one can easily compute the exact backward projection without any ambiguity. Finally, by applying a desired camera model over the 3D information (i.e., point cloud or mesh data in world space), the fisheye image is rendered using an optimized version of surface-fit algorithm [17]. The surface-fit algorithm is very similar to the process of rasterisation used in computer graphics. It is efficient and it operates at a high-computational speed. The method is notably local and requires only a small amount of memory (values of three vertices for each resampling position). The locality enables non-clashing memory handling and therefore, it can take advantage of parallelization.

Along with the rendered fisheye image, we store the generating angles θ and φ in a LUT. This angular information is then utilized in the rendering process of virtual views as explained in the next section.

Rendering virtual views

In Section 2, the proposed forward ray-base model is in theory efficient against high radial distortions and it is able to project rays from a wide FoV. However, the backward projection is given through LUTs containing all necessary angular information rather than via an approximated inverse function like in [11, 12]. In this section, we explore the advantages of using these LUTs to render new virtual views and we also propose an alternative real-time rendering pipeline.

Conventional rendering pipelines utilize the backward projection function to render new virtual views from fisheye imagery

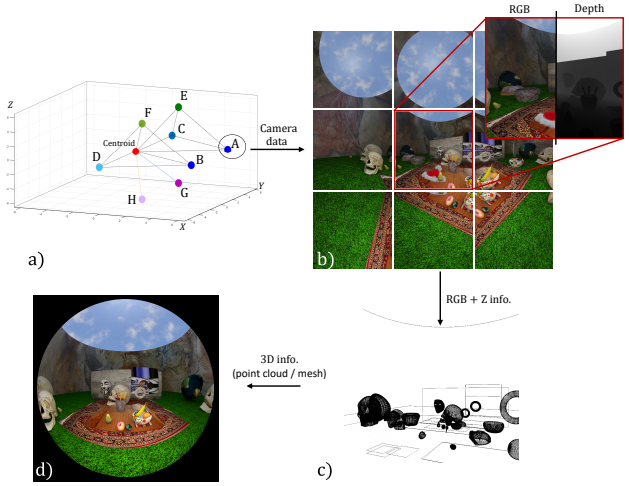


Figure 2. *Fisheye camera simulator pipeline. a) Stage 1: information about each camera {A, B, C, ..., H} in the camera rig; b) Stage 2: rendered perspective images (RGB + Z) using a third-party software; c) Stage 3: 3D structure of the scene represented as a mesh; d) Stage 4: Rendered fisheye image.*

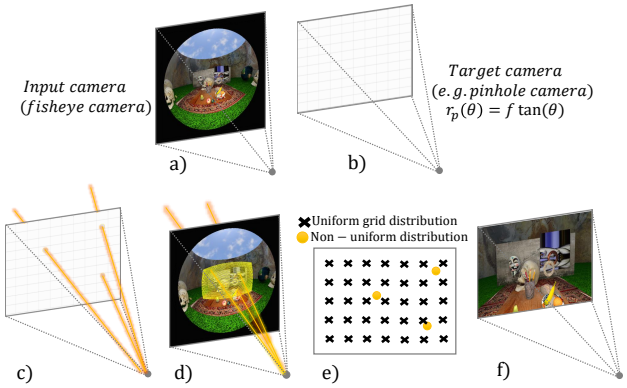


Figure 3. *Proposed rendering pipeline. a) and b) represent the input and output cameras, respectively. c) - f) are the four main steps used in the proposed rendering pipeline: c) target camera with respective ray information per pixel; d) forward projection of the rays onto the input camera; e) uniform to non-uniform image resampling; f) rendered image in the target camera.*

e.g., render panorama images. As mentioned in previous sections, the problem arises when the backward projection cannot be obtained analytically, which is the case of high-order polynomial functions. The use of numerical algorithms to compute its inverse results in two main problems. Firstly, it is an approximation which may have influence in the quality of the rendered virtual view. This error may be rather small if we fit a high-order polynomial function over the roots of the polynomial used in the forward projection. On the other hand, the error may be also significant if we try to fit a low-order polynomial function. Secondly, this is a computationally expensive operation because it must be performed on every single pixel of the fisheye image. This problem is mentioned but only partially discussed by the authors in [11, 12]. The authors in [12] developed a fast backward projection function by sacrificing its accuracy. Despite its significant speed-up, it is still far from real-time performance.

In 3DoF VR experiences, the panorama can be pre-computed using any of these offline methods mentioned above. Once the fisheye image is projected onto the panorama image then, virtual views can be rendered from it in real-time. This is a two-step approach that requires resampling the data two times and therefore, it can also lead to some artifacts in the final virtual view. In this paper, we are interested in analyzing a standard rendering pipeline where the virtual images are rendered directly from fisheye imagery.

The standard rendering pipeline starts by computing the backward projection mapping (compute the ray information for each pixel) or by utilizing LUTs to speed-up the process and achieve better accuracy. Then, the rays are forward projected onto the new virtual image grid. This gives origin to a non-uniform to uniform image resampling problem which is a challenging task and computationally expensive [18]. Instead of this procedure, we propose the rendering scheme illustrated in Figure 3.

Our proposal starts by defining the desired camera model of the new virtual camera (e.g., pinhole model). Then, for each pixel in the new virtual camera we compute the ray information. Since we know the direct inverse of the pinhole model (i.e., extraction of θ in Equation (1)) and the forward projection mapping of the fisheye camera, we can then forward project the data onto the fisheye image grid. The new virtual camera could also follow any other model as long as the forward and backward projection are known. The final step is to resample the data, a uniform to non-uniform image resampling problem using cubic or spline interpolation. Unlike in the standard pipeline, this rendering scheme requires less computational resources and therefore it can be implemented on CPU and still achieve real-time performance.

In sum, it is important to emphasize two factors in this section. First, LUTs can be used to speed-up the conversion of image points to rays and vice-versa when the model is rather complex. Second, the proposed pipeline avoids a non-uniform to uniform resampling problem and therefore, requires less computational resources while bringing better image quality as demonstrated next.

Experimental results

In this section, we present and discuss our experimental results including a comparison between the used ray-based model and other models from the literature, the performance of our rendering pipeline against the conventional rendering pipeline, and different output results that can be obtained with our virtual view rendering software.

The first set of experiments shown in Figure 4 compares the standard rendering pipeline with the proposed one. In terms of performance, the proposed pipeline was hundred times faster under the same test conditions. For instance, from a fisheye image with a 4k resolution and 195-degree FoV, a new virtual pinhole camera with 2K resolution and 90-degree FoV was rendered 150 times faster using the proposed pipeline. In terms of image quality, we also compared both rendered images using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) quality metrics as mentioned in the caption of Figure 4. This comparison was only possible to perform because we could access the 3D data structure of the scene, the same data used to generate fisheye images. The results show a significant improvement when our pipeline is used. As mentioned in the previous

Figure 4. *Visual comparison between different rendering pipelines. Indices 1 and 2 correspond to the first and second scenes. a): input fisheye image with 195° FoV for the first scene and 150° FoV for the second scene; b): ground-truth images with 90° FoV; c_1) standard rendering pipeline output with a PSNR-Y of 32,52 [dB] and SSIM of 0,92; d_1) proposed rendering pipeline output with a PSNR-Y of **35,00** [dB] and SSIM of **0,93**; c_2) standard rendering pipeline output with a PSNR-Y of 28,46 [dB] and SSIM of 0,75; d_2) proposed rendering pipeline output with a PSNR-Y of **30,50** [dB] and SSIM of **0,77**.*

section, conventional rendering pipelines face the process of non-uniform to uniform resampling which may also be limited due to the non-uniform distribution of the pixels information across the fisheye image. The information is denser around the fisheye image borders than in the center. In both pipelines, the output quality of the rendered virtual view is affected by the process of resampling. As shown in Figure 4, this process affects mostly the high frequencies of the signal and as a result, the output images have less details. For this particular experiment, our rendering pipeline outperformed the standard rendering pipeline in terms of computational speed and output image quality.

In our second experiment we have tested the model used in this paper against two other generic ray-based models that are widely used in calibration of fisheye lenses [11, 12]. For a direct comparison with the model described in [12], it was necessary to establish a connection between their backward projection and the incident angle θ used in our formulation. This relation is given by the the following equation:

$$\theta(r) = \frac{\pi}{2} - \arctan\left(\frac{f(r)}{r}\right), \tag{7}$$

where $f(r) = m_0 + m_2 r^2 + ... + m_n r^n$, $\{m_0, m_1 = 0, m_2, ..., m_n\}$ are the distortion coefficients, $r = \sqrt{u^2 + v^2}$ is the distance in pixels from the principal point to the projected point, and $n \in \mathbb{Z}_{\geq 0}$. Regarding the model described in [11], the forward projection model is given by Equation 2. Figure 5 shows the Mean Squared Error (MSE) for each model, i.e., the distance between four desired curves (pinhole, stereographic, equidistance, and equisolid angle projections), and the fitted ones, i.e., the proposed ray-based model and the two other generic ray-based models [11, 12]. For a fair comparison we have equalized the order of all polynomial functions used in these ray-based models. For instance, for a ninth order polynomial, the model in [12] requires about twice more calibration parameters than the model in [11]. Based on this experiment we can verify that there is no significant difference between the original ray-based model [11] and ours when

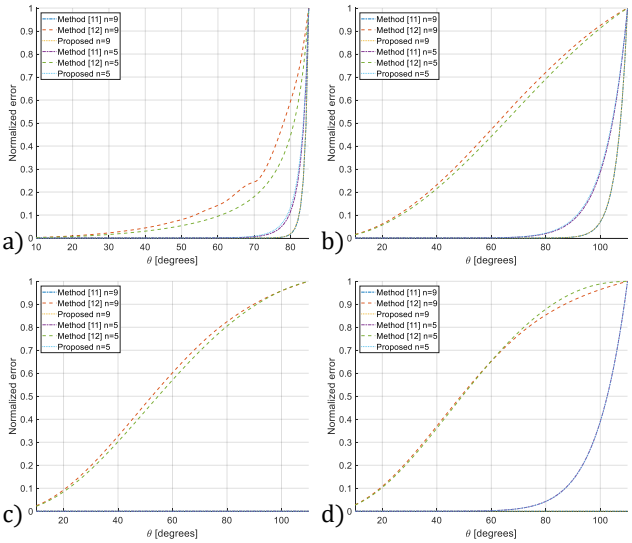


Figure 5. *Mean squared error to approximate: a) pinhole projection, $\theta \in [10, 85]°$; b) stereographic projection, $\theta \in [10, 110]°$; c) equidistance projection, $\theta \in [10, 110]°$; d) equisolid angle projection, $\theta \in [10, 110]°$.*

$m_0 = 1$. The original ray-based model in [12], that uses a fourth order polynomial, is also comparable with its own version using either a fifth or ninth order polynomial. From the calibration point of view, increasing the polynomial order to compensate from distortions may not justify the complexity required in the calibration process. Overall, for the same polynomial order, the ray-based model [11] outperforms the model [12] and, as expected, it also shows slightly better performance when $m_0 = 1$.

The last set of experiments shown in Figure 6 illustrates some of the properties of our rendering pipeline where one can change the output camera model or any intrinsic or extrinsic camera parameter online. For instance, one can change the target camera model, or tune any intrinsic or extrinsic parameters such as fo-

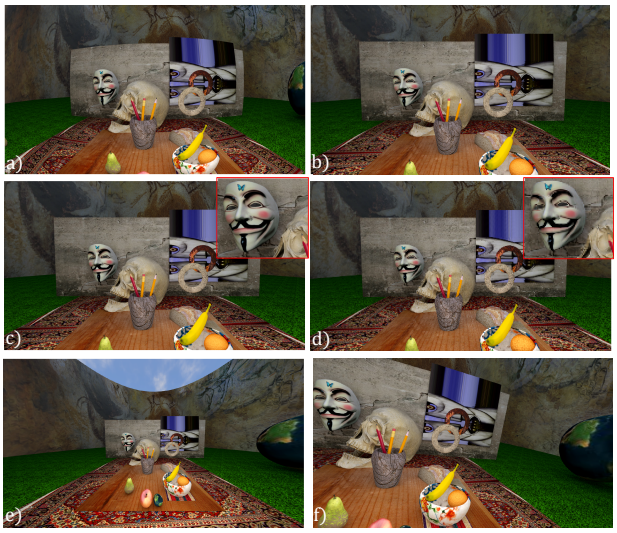


Figure 6. *Demonstration of different outputs from the proposed rendering pipeline. Different projection models: a) Equisolid angle; b) Pinhole. Pinhole model with: c) high resolution, d) low resolution, e) different focal length value, and f) different camera orientation.*

cal length, camera location or orientation, and camera resolution.

Conclusion

In this paper, along with the small details introduced in the generic camera model, our main contribution goes to the fast rendering pipeline of virtual views from images captured with cameras equipped with wide-angle lenses. While such optics offers a wide field-of-view, it also introduces severe distortions which have to be taken into account while rendering desired virtual views. Thus, one should use a proper camera model that supports any field-of-view value and different distortion magnitudes. However, only a strict group of camera models are able to tackle such problems. Furthermore, they are also difficult to manipulate due to their forward or backward projection functions. As a result, they often require procedures that may lead to inaccurate results or heavy computational burden. Nevertheless, thanks to look-up tables, we can speed-up any rendering task and ensure that there are no mismatches between the forward and backward projections. Finally, the proposed rendering pipeline is not only faster compared to conventional rendering pipelines, but it also provides more accurate results and better rendered images quality. Our procedures aimed at modelling fisheye optics and rendering virtual views could be adopted by any virtual reality system or 3D modeling software.

Acknowledgments

The work in this paper was funded from the European Unions Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 676401, European Training Network on Full Parallax Imaging.

References

[1] S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec, "A system for acquiring, processing, and rendering panoramic light field stills for virtual reality", SIGGRAPH Asia, 2018.

[2] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video", ACM Transactions on Graphics, vol. 34(4), pp. 1-13, 2015.

[3] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable inside-out image-based rendering", ACM Transactions on Graphics, vol. 35(6), 2016.

[4] R. Szeliski, "Image alignment and stitching: a tutorial", Foundations and Trends in Computer Graphics and Vision, vol. 2(1), pp. 1-104, 2006.

[5] J. Yu, "A light-field journey to virtual reality", IEEE MultiMedia, vol. 24(2), pp. 104112, 2017.

[6] S. Chan, H. Shum, and K. Ng, "Image-based rendering and synthesis", IEEE Signal Processing Magazine, vol. 24(6), pp. 22-33, 2007.

[7] R. Anderson, D. Gallup, J. Barron, J. Kontkanen, N. Snavely, C. Hernandez, S. Agarwal, and S. Seitz, "Jump: virtual reality video", ACM Transactions on Graphics, vol. 35(6), pp. 1-13, 2016.

[8] R. Hartley, and A. Zisserman, "Multiple view geometry in computer vision", 2nd Edition, Cambridge University Press, 2003.

[9] A. Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion", IEEE Computer Vision and Pattern Recognition, vol. 1, pp. 125-132, 2001.

[10] D. Claus, and A. Fitzgibbon, "A rational function lens distortion model for general cameras", IEEE Computer Vision and Pattern Recognition, vol. 1, pp. 213-219, 2005.

[11] J. Kannala, and S. Brandt, "A generic camera model and calibration method for conventional, wide-angle and fish-eye lenses", IEEE Pattern Analysis and Machine Intelligence, vol. 28(8), 2006.

[12] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for easy calibrating omnidirectional cameras", IEEE International Conference on Intelligent Robots and Systems, pp. 56955701, 2006.

[13] F. Steinicke, G. Bruder, K. Hinrichs, S. Kuhl, M. Lappe, and P. Willemsen, "Judgment of natural perspective projections in head-mounted display environments", ACM Symposium on Virtual Reality Software and Technology, pp. 35-42, 2009.

[14] Z. Tang, R. Grompone von Gioi, P. Monasse, and J. Morel, "A Precision Analysis of Camera Distortion Models", IEEE Image Processing, vol. 26(6), pp. 2694-2704, 2017.

[15] C. Ricolfe-Viala, and A. Sanchez-Salmeron, "Lens distortion models evaluation", Applied Optics, vol. 49(30), pp. 5914-5928, 2010.

[16] D. Brown, "Decentering distortion of lenses", Photogrammetric Engineering. vol. 32(3), pp. 444-462, 1966.

[17] A. Chuchvara, M. Georgiev, and A. Gotchev, "A speed-optimized RGB-Z capture system with improved de-noising capabilities", SPIE Image Processing: Algorithms and Systems XII, 2014.

[18] H. Sankaran, M. Georgiev, A. Gotchev, and K. Egiazarian, "Non-uniform to uniform image resampling utilizing a 2D farrow structure", Spectral Methods and Multirate Signal, pp. 3744, 2007.

Author Biography

Filipe Gama *is a PhD student at Tampere University. His research interests include 3D scene capture, virtual reality and light field imaging.*

Mihail Georgiev *is a researcher at Tampere University. His research interests include 3D scene capture, fusion and display technology.*

Atanas Gotchev *is a professor at Tampere University. His work concentrates on algorithms for multisensory 3D scene capture, transform-domain light-field reconstruction, and Fourier analysis of 3D displays.*

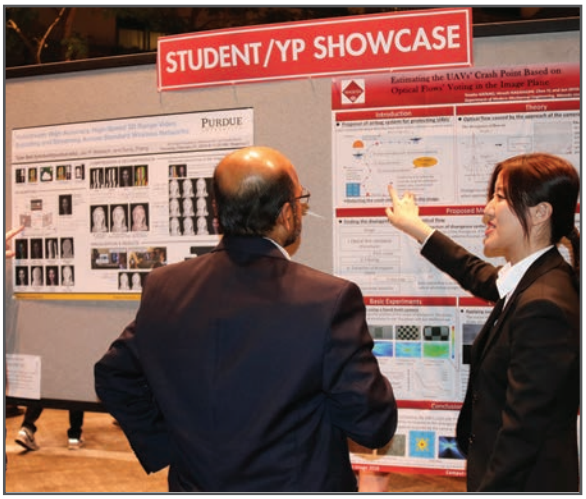
JOIN US AT THE NEXT EI!

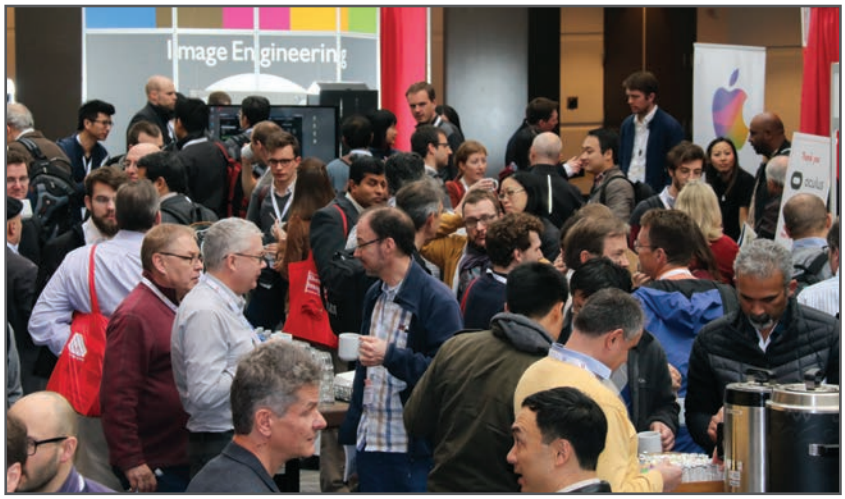
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org